useRef Hook:

What's useRef?

- 1. It created a reference and used in functional component.
- 2. Call useRef at the top level of your component to declare a ref.
- 3. It can be <u>used to access a DOM element directly</u>, means, generally we want to let React handle all DOM manipulation. But there are some instances where useRef can be used without causing issues.
- 4. It allows you to persist values between renders, means, if the state get changed or the component get re-render, the value stored with useRef doesn't change.
- 5. It is a React Hook that lets you reference a value that's not needed for rendering.

Implementation:

Calling const **reference** = useRef(initialValue) with the initial value returns a special object named **reference**. The reference object has a property current: you can use this property to <u>read</u> the reference value reference.current, or <u>update</u> reference.current = newValue.

When you change the <u>ref.current</u> property, React does not re-render your component. On the next renders, useRef will return the same object. React is not aware of when you change it because a ref is a plain JavaScript object.

> There are 2 rules to remember about references:

- 1. The value of the reference is *persisted* (remains unchanged) between component re-renderings.
- 2. Updating a reference doesn't trigger a component re-rendering.

Differences between refs and state:

refs	state
<pre>useRef(initialValue) returns { current: initialValue }</pre>	<pre>useState(initialValue) returns the current value of a state variable and a state setter function ([value, setValue])</pre>
Doesn't trigger re-render when you change it.	Triggers re-render when you change it.
Mutable—you can modify and update current 's value outside of the rendering process.	"Immutable"—you must use the state setting function to modify state variables to queue a re-render.
You shouldn't read (or write) the current value during rendering.	You can read state at any time. However, each render has its own snapshot of state which does not change.

Pitfall:

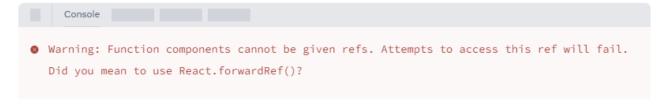
```
Reading or writing a ref during rendering breaks these expectations.
  function MyComponent() {
    // P Don't write a ref during rendering
   myRef.current = 123;
    // ...
   // P Don't read a ref during rendering
   return <h1>{myOtherRef.current}</h1>;
You can read or write refs from event handlers or effects instead.
  function MyComponent() {
   // ...
    useEffect(() => {
      // Vou can read or write refs in effects
    myRef.current = 123;
    });
    function handleClick() {
     // ✓ You can read or write refs in event handlers
   doSomething(myOtherRef.current);
    }
    // ...
If you have to read or write something during rendering, use state instead.
```

I can't get a ref to a custom component

If you try to pass a ref to your own component like this:

```
const inputRef = useRef(null);
return <MyInput ref={inputRef} />;
```

You might get an error in the console:



> Examples:

```
Click Counter:
import { useRef } from 'react';

export default function Counter() {
  let ref = useRef(0);

  function handleClick() {
    ref.current = ref.current + 1;
    alert('You clicked ' + ref.current + ' times!');
  }

return (
  <button onClick={handleClick}>
    Click me!
  </button>
);
}
```

Focusing a Text:

```
import { useRef } from 'react';
export default function Form() {
  const inputRef = useRef(null);

function handleClick() {
   inputRef.current.focus();
}

return (
  <>
      <input ref={inputRef} />
      <button onClick={handleClick}>
      Focus the input
      </button>
      </>);
}
```

Exposing a ref to your own component:

```
import { forwardRef, useRef } from 'react';

const MyInput = forwardRef((props, ref) => {
  return <input {...props} ref={ref} />;
});

export default function Form() {
  const inputRef = useRef(null);

function handleClick() {
  inputRef.current.focus();
}
```

Change the style of an element with useRef:

Changing input value with useRef:

- **O** Connect with Me:
- GitHub: https://github.com/priya42bagde
- LeetCode : https://leetcode.com/priya42bagde/
- YouTube Channel: https://youtube.com/channel/UCK1 Op30 pZ1zBs9l3HNyBw (Priya Frontend Vlogz)